数据分析 & PyTorch 图像多分类作业

By c7w

1 作业目标

本次作业是对学习的数据分析技能和人工神经网络知识的入门实践，其主体采用**代码填空**的形式进行。你需要按照本实验指导中的流程，尝试阅读并理解现有的实验框架，然后在现有实验框架的基础上做代码补全，以实现对应子任务所要求的功能。

我们本次作业的最终目标是要利用预先获取到的风景图像数据，训练一个固定图像分辨率的风景图像**多分类**网络。我们会对于给定的一张固定分辨率的图片，预测其中有无山脉、有无水流、有无天空，以实现对其自动打标签的效果。风景图片数据我们这里通过清华云盘链接的形式给出，请各位同学下载后，按照 SubTask 0 中的要求解压到指定位置。

具体而言，在本次作业中我们要实现以下内容：

- SubTask 0：环境配置与安装（15 p.t.s）
- SubTask 1：数据预处理（45 p.t.s）
- SubTask 2：训练框架搭建（60 p.t.s）
- SubTask 3：结果提交（30 p.t.s）
- SubTask 4：代码整理与开源（Bonus. 10 p.t.s）

2 环境配置与安装（15 p.t.s）

2.1 准备 `Python` 环境（10 p.t.s）

我们在前面的课程已经学习过 `conda` 环境管理器的使用，你应该可以理解下面指令的作用。

```
conda create -n ai python=3.8
conda activate ai
pip install -r requirements.txt
```

如果你是 NVIDIA 显卡的受害者，那么恭喜你可以使用 CUDA 加速的相关库。你可以理解成你可以充分利用你显卡的算力来做并行计算。一张显存大于等于 4 GB 的显卡就非常符合本次任务的要求：请删除 `requirements.txt` 中 `torch==1.12.0` 这一行，然后安装带有 CUDA 加速版本的 `torch`：

```
# Windows / Linux
conda install pytorch torchvision cudatoolkit=11.3 -c pytorch
# macOS is not supported yet for CUDA :(
# Link copied from https://pytorch.org/
```

如果没有符合上述要求的显卡也没关系，你的 CPU 和风扇已经做好了煎鸡蛋的准备。经测试，未经 CUDA 加速的机器使用 CPU 训练一轮大概需要 30 ~ 60 min，这时间在 CUDA 加速后缩短为 6 ~ 10 min。我们默认选取训练 10 轮，你可以在后续的环节进行配置调整。

2.2 准备数据集（5 p.t.s）

请从上面清华云盘的链接中下载数据集，然后解压到 `data` 目录下，保证 `data` 目录下直接存在 `train`、`val`、`test` 文件夹与 `LICENSE` 文件。

请阅读 LICENSE 文件，继续进行本作业代表你已知晓并同意 LICENSE 文件中的所有内容。

3 数据预处理（45 p.t.s）

在这一部分我们需要撰写数据预处理的相关函数，你可能会用到 `Pillow`、`NumPy` 等库。

我们首先来讲解 `./data` 下的文件。`imgs` 中图片的**逐像素**标注位于 `train/labels` 下的同名文件中。你可以将每张逐像素标注认为是一张灰度图，存储了 `0-255` 这 256 个数的其中之一。标签 ID 与标签的对应关系如下：

标签 **ID**	标签类别
0, 7	Mountain
1	Sky
2, 3, 8, 16, 20	Water

因为这些数值都非常接近 0（表征黑色），所以你肉眼无法区分像素点所代表的类别。

我们的目的是，对于一张 `imgs` 下的图片，读取其在 `labels` 下的对应标注：我们要判断原图中有没有山、有没有天空、有没有水，以此来实现对图片打“标签”的效果。接下来我们便要对这些图片及其标签进行预处理，其步骤为：

对于一张图片，如果其中标记为 “Mountain” 像素的个数**超过**了总像素的 20%，我们就认为这张图片中含有 “Mountain”。同理，如果一张图片中标记为 “Sky”、“Water” 的像素个数超过了总像素个数的 20%，我们就认为这张图片中含有 “Sky”、“Water”。

3.1 数据预处理（45 p.t.s）

接下来请阅读并补全 `datasets/dataset_landscape_generator.py` 中的代码，以达到可以产生与 `data/val/file.txt` 相类似的 `data/train/file.txt` 的效果。为达成目标，你只需要修改 `# TODO Start #` 与 `# TODO End #` 之间的内容。

在你完成这部分后，你应该在 `./data/train` 下生成了一个 `file.txt` 文件。你可以与这个文件作对比以查看中间结果是否正确。

【任务清单】

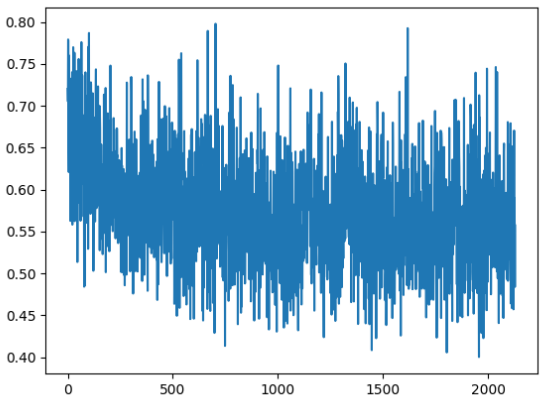
- 阅读 `data/val/file.txt`，了解我们要处理成的目标数据格式（5 p.t.s）
- 在 `process_data` 函数中：（20 p.t.s）
  - 构建 `image_dir` 与 `label_dir`（`pathlib.Path`）
  - 构建 `filename_list`（`os.listdir`，对字符串切片取消后缀名）
  - 将处理后的字符串写入 `{working_dir}/file.txt`（写入文件）
- 在 `calc_label` 函数中：（20 p.t.s）
  - 按照函数注释中的内容完成函数
  - 正确的做法预处理 `train` 数据集中的数据应该在 10 min 以内完成
  - 提示：将 `label` 视为传统的二维数组进行遍历需要至少 1024×768 次计算，而这在 8000 张图片上运行会导致处理效率的降低
  - 提示：能否借助 NumPy 中的工具进行高效并行计算?
  - 提示：`np.isin`，`np.sum`

4 训练框架搭建（60 p.t.s）

在这里我们会体验一次传统的 PyTorch 神经网络训练框架的搭建过程。这不仅仅是一个应用性的研究课题，也是一个工程项目，因此在编写时请注意你的项目规范。接下来我们继续细分成不同子任务来完成这个搭建流程。

4.1 工具函数撰写（10 p.t.s）

在这部分我们需要用到 `matplotlib` 中的有关绘制折线图的工具，撰写 `./utils/metric.py` 中的 `draw_loss_curve` 函数。你绘制的 x 坐标轴应该是 `loss_list` 列表的下标，y 坐标轴为该列表中元素的值。请根据函数注释补全有关内容。你最终绘制的折线图应类似于：



【任务清单】

- 在 `utils/metric.py` 中：

- 完成 `draw_loss_curve` 函数的撰写

4.2 数据集与数据加载器（15 p.t.s）

在这部分我们需要理解 PyTorch 对训练数据的这一层抽象。我们会接触到两个常用类，`torch.utils.data.DataLoader` 与 `torch.utils.data.Dataset`。你可能需要查阅官方文档它们应该如何构造。

我们首先来介绍后者。我们要构建的数据集类 `datasets/dataset_landscape.py` 应该是这个 `torch.utils.data.Dataset` 类的子类。然后，其至少应该实现三个方法：`__init__` 方法传入数据集对象构造时用的配置，修改一些内部成员的值以记录当前数据集对象的状态。`__len__` 方法返回当前数据集对象的大小 L。`__getitem__` 方法接受一个在 $[0, L)$ 之间的整数 K 并返回数据集中的第 K 个元素。以训练集对应的数据集对象为例，其 `__init__` 方法应建立并记录一个 $[0, 8000)$ 与训练集中所有图片与标注的对应关系，`__len__` 方法返回 8000，`__getitem__` 方法在调用时返回指定下标的图片和标签，以 `torch.Tensor` 的形式。在处理一些尺寸较小的数据时，比如 `label` 只用了 3 个 bool 型，可以存储在内存中加速读取，而对于规模较大的数据，比如一张 1024×768 的图片，应在后续获取时再将其打开，以节省内存开销。

然后我们来介绍前者。`DataLoader` 接收一个 `Dataset` 对象作为必选参数，主要用来控制训练过程中如何利用数据集对象这一行为。比如一次从数据集中采样多少个，比如是否将数据集随机打散等等。

【任务清单】

- 在 `datasets/dataset_landscape.py` 中：
  - 修改我们的数据集继承 `torch.utils.data.Dataset`（类继承的写法）
  - 试着理解 `__init__` 函数，这里做了什么？（列表产生式）
  - 在 `__len__` 中返回当前数据集的大小（哪个变量的大小代表了当前数据集的大小？）
  - 在 `__getitem__` 中使用 `PIL` 读取图片并放缩到指定尺寸（Pillow）
  - 理解 `__getitem__` 中 `numpy.transpose` 的用法后，取消注释该行
- 在 `utils/experiment.py` 中：
  - 仿照 `val_dataloader` 的写法，使用 `dataset` 对象构建 `train` 下的 `dataloader`，注意打开 `random shuffle` 功能
  - 上网查阅有关资料，这里 `num_workers` 配置的作用是什么？

4.3 模型定义（10 p.t.s）

我们使用深度卷积神经网络 VGG-16 `VGG16` 作为特征提取器，将提取到的特征向量最终再经过 3 个不同分类头 `ClassificationHead` 分别对应我们三个标签，每个分类头的输出结果是一个二分类，代表有该标签或无该标签。

每个模型类需要继承自 `torch.nn.Module` 类，且至少需要实现 `__init__` 函数与 `forward` 函数，前者是构造函数，传入一些参数，构造其子模块，后者则表示模型该如何将子模块之间串接起来，由输入组装输出。

【任务清单】

在这一阶段功能已为你全部实现好，但是你需要阅读并理解 `./models/MultiClassificationModel` 中的内容，并完成以下任务：

- `make_layers` 方法与 `VGG16` 类：
  - 上网查阅资料，了解什么是工厂方法。
  - `nn.Sequential` 是什么？它是否继承自 `torch.nn.Module`？如果让你自己实现这个类，它的 `__init__` 与 `forward` 函数应该如何定义？
  - `make_layers` 的返回值中 `*layers` 的 `*` 是什么含义？
- `ClassificationHead` 类：
  - `nn.ModuleList` 是什么？它跟普通的 List 有什么区别？
  - `nn.Dropout` 的作用是什么？
- `MultiClassificationModel` 类：
  - 这个模型的参数量有多大？
  - `forward` 函数中每一步的输入输出的张量形状分别是什么？
  - 提示：你通过在 forward 函数中使用 `IPython.embed()` 打断点的形式回答上述问题

4.4 主逻辑实现（20 p.t.s）

接下来我们便开始查看程序的主运行逻辑，程序的入口是 `main.py`。在你完成下面的代码补全和思考之后，便可以开始训练模型了。

【任务清单】

- 在 `./main.py` 中：
  - **运行时参数解析**
    - 仿照 `args.save_freq` 定义 `args.val_freq` 与 `args.print_freq` 两个变量，类型均为 `int`，默认初值均为 `1`
    - 使用 `IPython.embed()` 断点查看，`args` 变量里都有什么？
  - **优化器定义**
    - 查阅官方文档了解 Adam 优化器的定义方法，并将其实现
  - **损失函数定义**
    - 查阅官方文档，了解 `nn.CrossEntropyLoss()` 的使用方法
- 在 `./utils/experiment.py` 中：
  - 在 `initiate_environment` 函数中：

    - 思考：我们设置随机数种子的目的是什么？
  - 在 `save_model` / `load_model` 函数中：
    - 思考：`torch.save` 和 `torch.load` 的对象是什么数据类型？
    - 补全 `optimizer.load_state_dict` 函数
  - 在 `train_one_epoch` 函数中：
    - 注释掉的几行代码分别在做什么？理解后取消代码注释
  - 在 `evaluate_one_epoch` 函数中：
    - 思考：`with torch.no_grad()` 有什么作用？
    - `calc_accuracy` 指向的函数在哪？它是怎么实现的？
- 思考：该传入怎样的参数开始一次训练？该传入怎样的参数进行一次测试？

5 结果提交（30 p.t.s）

一般我们不会检查前面几项你的完成情况，一旦你提交了最终结果前面的项目即记为满分。请复制你测试出来的 `.txt` 文件，提交到 http://121.5.165.232:14000 中刷新页面。为防止通过多次提交恶意获取测试集答案，我们对测试结果进行了四舍五入后返回。为方便我们统计大家的参与情况，后续发放服务器作为奖励，请正确填写自己的学号。具体评分标准我们这里不予公开，但正确运行的代码应至少在本项目中取得 80% 及以上的分数。

参数调优是一个经久不衰的话题：在上述流程中，有哪些参数可以供我们调整，使得模型可能达到更好的效果？你可以多做尝试，甚至可以大胆地更换模型架构，或是加载其它预训练过的模型作为基础，你只需要在最终的报告中写明你的做法即可 :)

6 代码整理与开源（Bonus. 10 p.t.s）

作为我们的 Bonus 评定内容，你可以在完成作业后，将代码进行托管，如 GitHub， Tsinghua Git 等，然后在原仓库中新建 Issue，提交代码仓库地址。请整理你的代码，留足充分的注释后，向大家说明你的代码的使用方式。在向远程仓库推送文件时注意不要提交数据集与你的模型存档点，存档点的合理公开方式应该是放在云盘中并分享下载链接。

此外，如果你没有提交最终结果，想获得部分分数的认定，你也应该通过这种渠道进行申请。如有疑问请联系负责人 cc7w@foxmail.com，微信号 c7wc7w。

7 Appendix

7.1 标签映射表

```
00 "mountain", "sky", "water", "sea", "rock"
05 "tree", "earth", "hill", "river", "sand"
10 "land", "building", "grass", "plant", "person",
15 "boat", "waterfall", "wall", "pier", "path",
20 "lake", "bridge", "field", "road", "railing",
25 "fence", "ship", "house", "other"
```